

# Case Study:

## Document Capture and Indexing For SharePoint

# GOSCAN<sup>®</sup>



# Presenters



- Paul Smietan
  - CTO, GoScan, Inc.
  - Former Microsoft senior consultant
  - Pursuing Ph.D. in computer science at Caltech
- Mike Stuhley
  - President, GoScan, Inc.
  - Former Scantron and Cardiff executive
  - No chance of getting into Caltech

# Customer Pain

- Fast growing healthcare organization
  - Improved Information Access
  - Better workflow
  - Support EMR implementation
  - Enforce consistency in procedures across departments



# Why MOSS?

- Financial stability
- Product breadth
- License cost
- Customization ability
- Eforms
- MS Office Integration
- Programmer availability



# Roadblock 1



# How Do We Feed MOSS?

- Wanted to scan from multiple departments and locations into SharePoint
- Needed manual and automatic indexing
- Has to be easy, fast, accurate and inexpensive



# No Rx From Microsoft

- Microsoft doesn't have a scanning solution
- Third party tools need to pick up the slack



# Software Requirements

- Simple
- Smart
- Affordable



# Simple



- One simple interface
- Minimal user skills required
- Use on “every desktop, every day”

Just click on the green button



# Smart

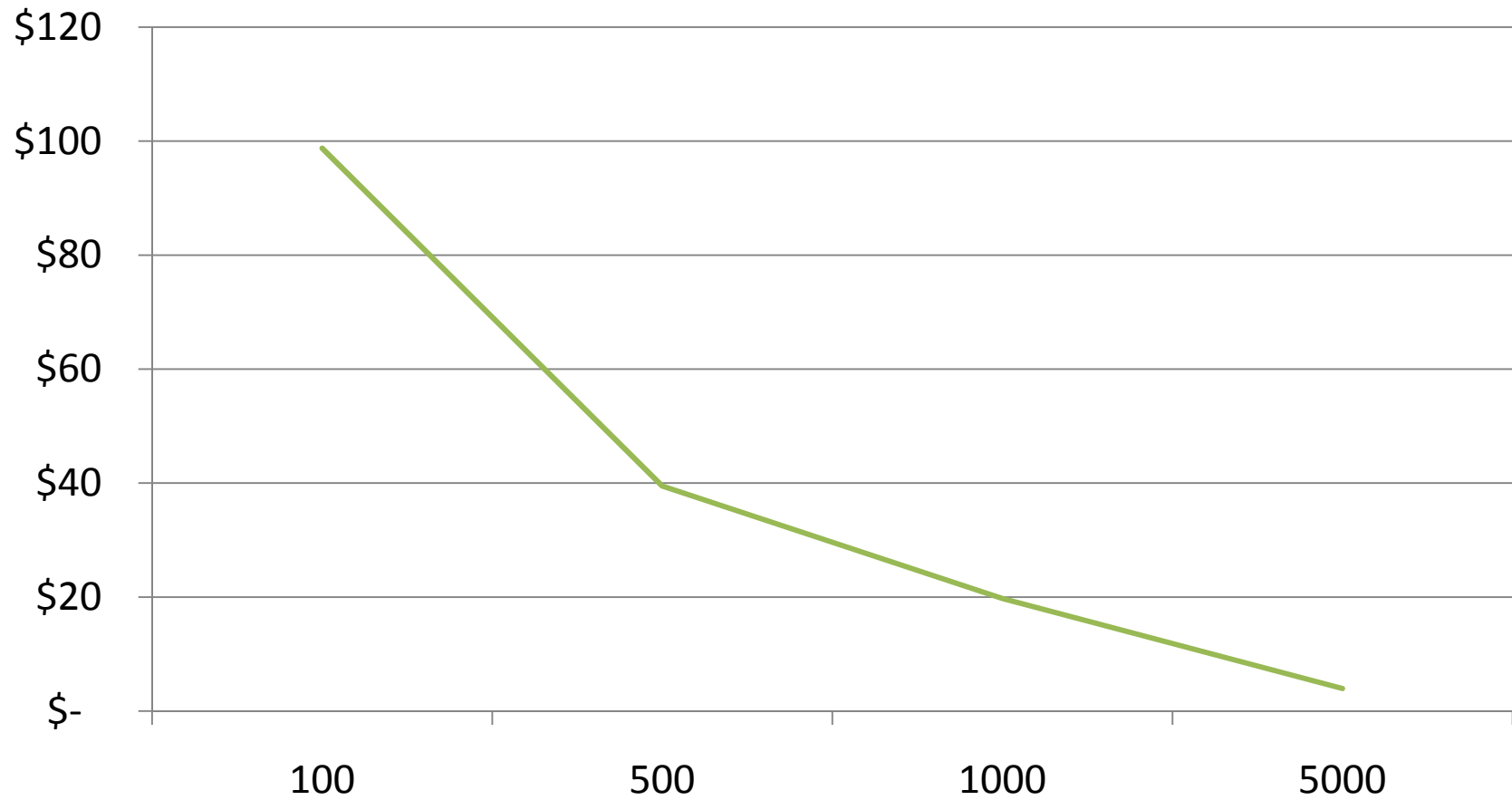


- Automatic barcode or OCR filing
- Index paper or electronic documents
- Users key in 2 fields and then hit save



Field Name	Value
MemberID	12345
DocumentCategory	Labs

# Cost Per User



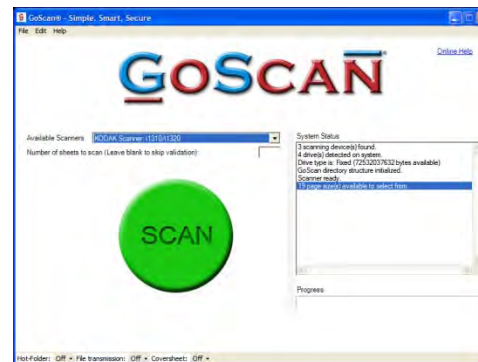
# Other Requirements

- Compatible with any scanner
- Unlimited scanning (no per page costs)
- Works for central or distributed scanning
- Export to other targets besides MOSS

Scan



Process



Store



**GOSCAN**



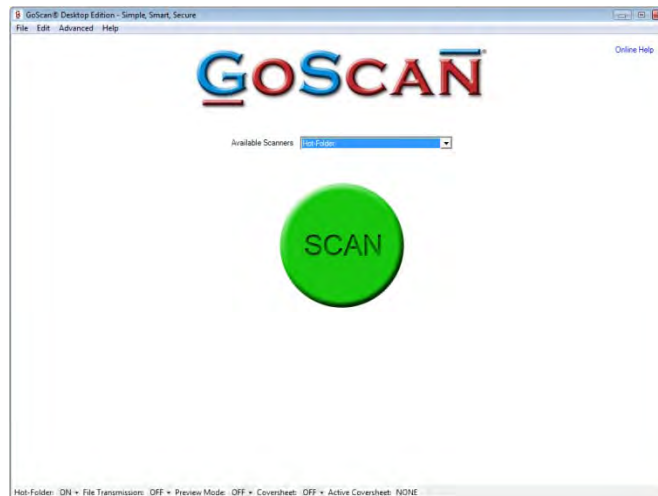
# Diverse Filing Requirements

- Scan paper without indexing
- Scan paper with barcodes
- Scan with barcode coversheets
- Scan with electronic coversheets
- File existing electronic documents
- Need one solution for everyone



# Scan Without Index

- Get paper from point A to point B with a simple, consistent interface
- Target can be a directory, FTP site, web server, SQL, MOSS or Dynamics
- Indexing done at a different stage



# Scan Barcode Without Coversheet

- Single page with any barcode
- Read barcode to identify page
- Output image only (with specific name) or image with XML



# Scan With Barcode Coversheet

- Coversheet with barcode goes on top or in between multiple pages
- Read barcode to identify document
- Output image only (with specific name) or image with XML

Create 10 bar coded coversheets with Microsoft Word	<p>SCANNABLE COVER SHEET</p> <p>Customer Number</p>  <p>12345</p>
Insert coversheets into stack of 30 documents	
Scan 40 pages with GoScan	
GoScan automatically creates 10 image files	


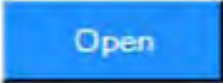

# Scan With Electronic Coversheet

- Type in index values
  - Use optional database lookup to populate fields
- Press scan



Field Name	Value
MemberID	12345
DocumentCategory	Labs

# File Existing Electronic Document

Save electronic document							
Click open with GoScan	<p>Images</p>  						
Type in index values	<table border="1"><thead><tr><th>Field Name</th><th>Value</th></tr></thead><tbody><tr><td>MemberID</td><td>12345</td></tr><tr><td>DocumentCategory</td><td>Labs</td></tr></tbody></table>	Field Name	Value	MemberID	12345	DocumentCategory	Labs
Field Name	Value						
MemberID	12345						
DocumentCategory	Labs						
GoScan automatically sends indexed document into desired repository	<ul style="list-style-type: none"><li><input type="radio"/> SharePoint</li><li><input type="radio"/> SQL Server</li><li><input type="radio"/> Oracle Server</li><li><input type="radio"/> IBM DB2</li></ul>						

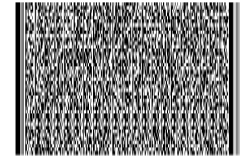
# Forms Processing and SharePoint

- Hand print (ICR)
- Machine print (OCR)
- Check marks (OMR)
- Bar codes
  - 1 and 2 dimensional

M	I	K	E	
---	---	---	---	--

4563418
---------

Yes     No



# Technical Hurdles

- Search and “reusable” documents
- What image format to use
- Auto populating indexes
- Connecting to MOSS



# Enabling Enterprise Search

- Basic function of putting documents into MOSS is to find them again
- Full text OCR adds word search to metadata search
- Full text OCR can create a “reusable” Word file or a “searchable” PDF+Text file



# Image Formats For SharePoint

- PDF
  - Universally loved
  - Can hold image and text
- TIF
  - Basic building block of imaging systems
  - Needs an application to display images
- JPG
  - Displays natively in browser
  - Single page only



# Connect to Index

- Easily grab additional metadata from SQL
- E.g., “ssn” gets name, phone, etc.

Server Definition Parameters

Server(s) Found: 4  
PAULIE

Database(s)  
GoScan1

Tables found: 01  
ssnTable

Fields found: 02  
ssn

User ID:  
[ ]

Password:  
[ ]

Primary Key Field  
[ ]

Use Custom External Search Query

select b.ssn, a.LastName, a.FirstName from GoScan1.dbo.namesTable a inner



Field Name	Value	Look Up Enabled	Look Up
ssn		<input checked="" type="checkbox"/>	
		<input type="checkbox"/>	

# How Does Auto Populate Work?

## Main User Interface

- Perform basic look-up to each key field(s)
- Perform custom search look-up to field(s) that is more complex
- Perform exclusive search look-up to field(s) autonomous of other field(s)

## Detailed Search

- User presses F2 key to perform a detailed search of field(s).
- Easily view and select a record that may contain multiple occurrences of a common name.
- Select the record and data is transferred back to main user interface.

## SCAN

- User inserts paper into scanner and presses the green SCAN button, or:
- User selects an electronic image located on the local disk or mapped network drive and then saves their work.

# GOSCAN®

Available Scanners



### Entry Grid

- 
- 

### Images

- 
- 

Field Name	Value	<input type="checkbox"/>	Look Up
MemberID		<input type="checkbox"/>	
MemberFirstName	Steve	<input checked="" type="checkbox"/>	
MemberLastName	Ballmer	<input checked="" type="checkbox"/>	Ballmer
MemberDOB		<input type="checkbox"/>	
DocumentCategory	Doctor	<input checked="" type="checkbox"/>	Doctor
DocumentType	DoctorType-3	<input checked="" type="checkbox"/>	DoctorType-3
RecordBeginDate		<input type="checkbox"/>	
RecordEndDate		<input type="checkbox"/>	
DocumentNote	CEO	<input checked="" type="checkbox"/>	

Detailed Search

MemberID	MemberFirstName	MemberLastName	MemberDOB
	<b>Steve</b>		
1234	Steve	Smith	1/1/1960
2323	Steve	Jones	3/3/1934
3456	Steve	Ramirez	4/3/1970
4541	Steve	Ramos	7/13/2000
9898	Steve	Smyth	2/5/1980
3322	Steve	Ballmer	7/5/1950
5555	Steve	Shapiro	12/1/1985

Entry Grid

Clear Values

Detailed Search (F2)

Open

Save

Field Name

- MemberDOB
- DocumentCategory
- DocumentType
- RecordBeginDate
- RecordEndDate
- DocumentNote
- MemberID
- MemberFirstName
- MemberLastName

Look Up

Ok

Cancel

# Connection Details

- SSIS, web services, bulk import
- GoScan streams xml with or without the image files embedded (pdf/tif/jpg)
- Connect for both indexing and output

Post Methodology

Use GoScan HTTP Post method

Use Custom Web Service method



# Connect to export

- Easily call your own custom web method
  - Select “custom web service method” in GoScan
  - Name your web service “Upload”
  - Supply username, password, domain

Post Methodology

- Use GoScan HTTP Post method
- Use Custom Web Service method

# Customer Pain

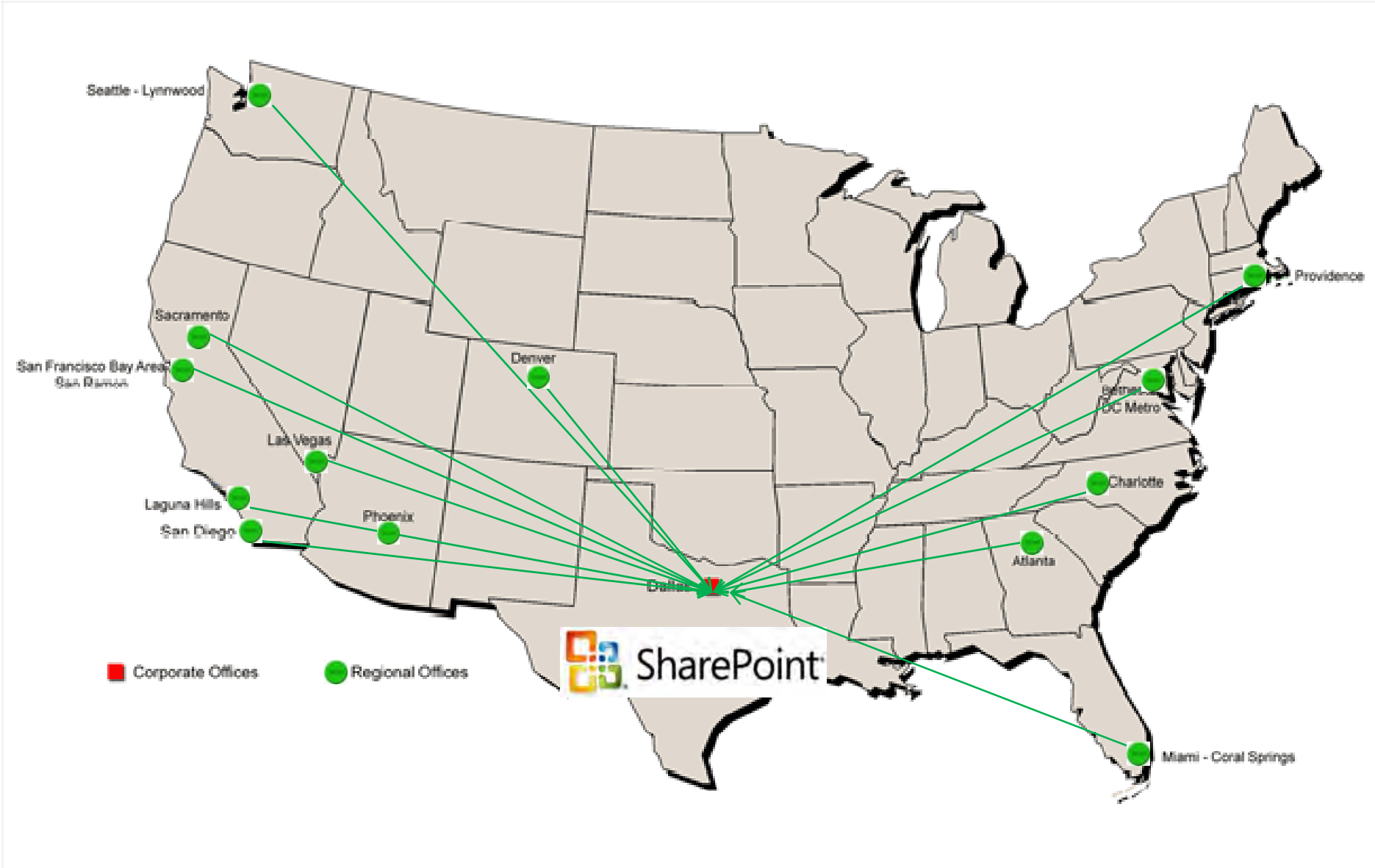
- National real estate company
  - Get documents from 2,500 people in 40 geographies
  - Regulatory issues
  - Better workflow
  - Enforce consistency in procedures across offices



# Why MOSS?

- Financial stability
- Product breadth
- License cost
- Customization ability
- Eforms
- MS Office Integration
- Programmer availability





# Roadblock 1



# How Do We Feed MOSS?

- Wanted to scan from thousands of locations into SharePoint
- Needed manual and automatic indexing
- Has to be easy, fast, accurate and inexpensive



# No Rx From Microsoft

- Microsoft doesn't have a scanning solution
- Third party tools need to pick up the slack



# Software Requirements

- Simple
- Smart
- Affordable



# Simple



- One simple interface
- Minimal user skills required
- Use on “every desktop, every day”

Just click on the green button



# Smart

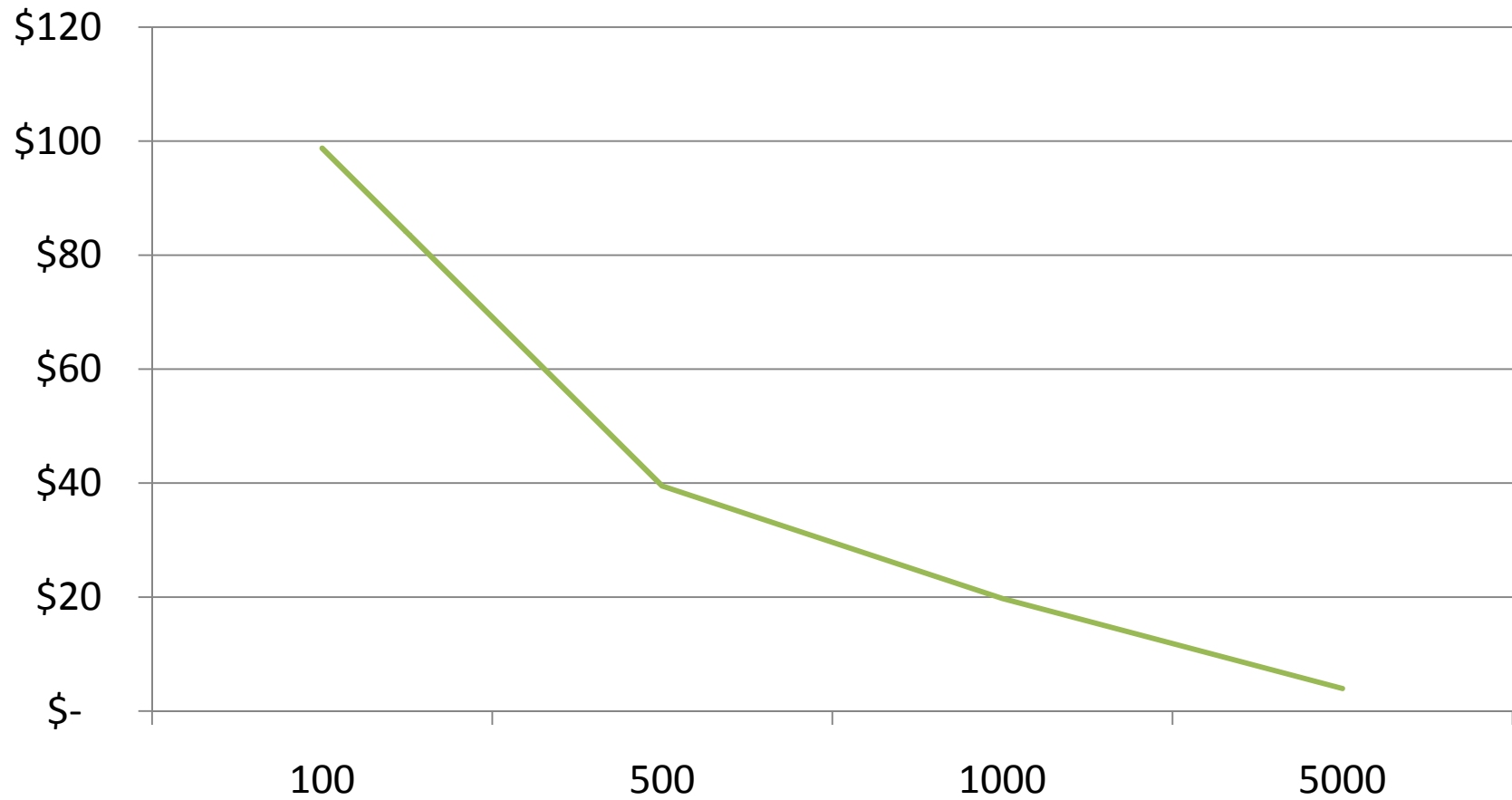


- Automatic barcode or OCR filing
- Index paper or electronic documents
- Users key in 2 fields and then hit save



Field Name	Value
MemberID	12345
DocumentCategory	Labs

# Cost Per User



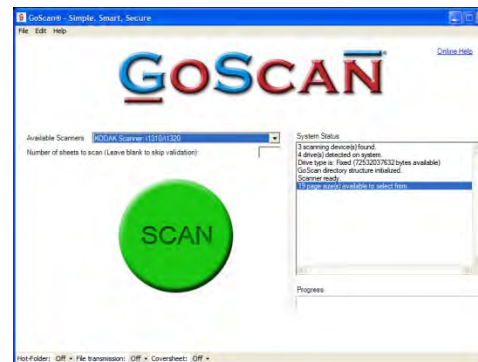
# Other Requirements

- Compatible with any scanner
- Unlimited scanning (no per page costs)
- Works for central or distributed scanning
- Export to other targets besides MOSS

Scan



Process



Store



**GOSCAN**



# Technical Overview

- Implementing SharePoint without outside help
- GoScan provided image and xml metadata but they did not know how to process this



# GoScan To The Rescue

- GoScan provided a sample project to help them create custom web methods so that data can be processed and sent to the correct destination



# Custom Web Service (upload)

Service.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.Services;
//using Microsoft.SharePoint;
//using Microsoft.SharePoint.Utilities;
using System.IO;

[WebService(Namespace="http://tempuri.org/")]
[WebServiceBinding(ConformsTo=WSIProfiles.BasicProfile1_1)]
// To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.
// [System.Web.Script.Services.ScriptService]
public class Service : System.Web.Services.WebService
{
    public Service()
    {
        //Uncomment the following line if using designed components
        //InitializeComponent();
    }

    [WebMethod]
    //public string GetSiteListCount()
    //{
    //    SPWeb myWebSite = SPContext.Current.Web;
    //    SPListCollection lists = myWebSite.Lists;

    //    return (myWebSite.Title + " contains " + lists.Count.ToString() +
    //    " lists.");
    //}

    [WebMethod]
    public void Upload( String goScanXML )
    {
        // Insert your logic here!
    }
}
```

Want to do SharePoint dev? Uncomment these two lines!

Add your logic to the Upload() method to meet your specific needs.

Ready Ln 25 Col 8 Ch 8 IN5

# Change the URL and Deploy

The screenshot displays the Microsoft Visual Studio IDE with the following components:

- Code Editor:** Shows the `Service.cs` file. The code includes using statements for `System`, `System.ComponentModel`, `System.Diagnostics`, `System.Web.Services`, `System.Web.Services.Protocols`, and `System.Xml.Serialization`. It defines a `Service` class that inherits from `SoapHttpClientProtocol`. The `Service` class has a `Service()` constructor where the `this.Url` property is set to `http://moss.goscan.com:16674/CustomUpload/Service.asmx`. Other methods include `Upload`, `BeginUpload`, `EndUpload`, and `UploadAsync`.
- Properties Window:** Located on the right side of the code editor, it shows the `Service` class's properties, with the `Url` property highlighted.
- Solution Explorer:** Located on the far right, it shows the project structure for `CustomUpload`, including `App_Code`, `App_Data`, `Service.asmx`, `Service.cs`, and `web.config`.
- Red Arrow:** A large red arrow points from the text "Change the URI to point to your web server and deploy!" to the `this.Url` property in the `Service()` constructor.

At the bottom of the IDE, the status bar shows "Ready", "Ln 32", "Col 5", "Ch 5", and "INVS".

# WSDL – Function Prototypes

The screenshot shows the Visual Studio IDE with a C# file named `Service.cs` open. The code is auto-generated by WSDL and includes several annotations and a service class definition. A yellow sticky note is placed over the code, and a red arrow points to the `this.Uri` property in the `Service` constructor.

```
// <auto-generated>
// This code was generated by a tool.
// Runtime Version:2.0.50727.3002
//
// Changes to this file may cause incorrect behavior and will be lost if
// the code is regenerated.
// </auto-generated>
//

using System;
using System.ComponentModel;
using System.Diagnostics;
using System.Web.Services;
using System.Web.Services.Protocols;
using System.Xml.Serialization;

//
// This source code was auto-generated by wsdl, Version=2.0.50727.3038.
//

/// <remarks/>
[System.CodeDom.Compiler.GeneratedCodeAttribute( "wsdl", "2.0.50727.3038" )]
[System.Diagnostics.DebuggerStepThroughAttribute()]
[System.ComponentModel.DesignerCategoryAttribute( "code" )]
[System.Web.Services.WebServiceBindingAttribute( Name="ServiceSoap", Namespace="http://tempuri.org/" )]
public partial class Service : System.Web.Services.Protocols.SoapHttpClientProtocol
{
    private System.Threading.SendOrPostCallback UploadOperationCompleted;

    /// <remarks/>
    public Service()
    {
        this.Uri = "http://localhost:3970/CustomUpload/Service.asmx";

        /// <remarks/>
        public event UploadCompletedEventHandler UploadCompleted;

        /// <remarks/>
        [System.Web.Services.Protocols.SoapDocumentMethodAttribute( "http://tempuri.org/Upload", RequestNamespace="http://tempuri.org/", ResponseNamespace="
        public void Upload( string goScanXML, string pathFQN )
        {
            this.Invoke( "Upload", new object[] {
                goScanXML,
                pathFQN } );
        }

        /// <remarks/>
        public System.IAsyncResult BeginUpload( string goScanXML, string pathFQN, System.AsyncCallback callback, object asyncState )
        {
            return this.BeginInvoke( "Upload", new object[] {
                goScanXML,
                pathFQN }, callback, asyncState );
        }

        /// <remarks/>
        public void EndUpload( System.IAsyncResult asyncResult )
        {
            this.EndInvoke( asyncResult );
        }
    }
}
```

Don't forget to change the temporary URI when you go to production phase!

Simply point to your IP/Server here!  
Ex. <http://ProdSvr-1/CustomUpload/Service.asmx>

# Update your WSDL!

```
Visual Studio 2008 Command Prompt
Directory of D:\Corporate\Projects\GoScan\CustomUpload
07/15/2009  09:42    <DIR>          .
07/15/2009  09:42    <DIR>          ..
07/15/2009  09:42    <DIR>          App_Code
07/30/2008  07:54                87 Service.asmx
07/14/2009  20:24            3,532 Service.cs
07/14/2009  12:48            8,690 web.config
           3 File(s)          12,309 bytes
           3 Dir(s)      277,903,073,280 bytes free

D:\Corporate\Projects\GoScan\CustomUpload>wsdl http://moss.goscan.com:16674/Cust
omUpload/Service.asmx
Microsoft (R) Web Services Description Language Utility
[Microsoft (R) .NET Framework, Version 2.0.50727.3038]
Copyright (C) Microsoft Corporation. All rights reserved.
Writing file 'D:\Corporate\Projects\GoScan\CustomUpload\Service.cs'.

D:\Corporate\Projects\GoScan\CustomUpload>
```

# Use Web Publishing To Deploy

CustomUpload - Microsoft Visual Studio

File Edit View Website Build Debug Tools Test Window Help

Debug Any CPU ExportConnectors.PostUsingWebs

Copy Web D:\...CustomUpload\ Service.asmx App\_Code\Service.cs Service.cs

Connections: V:\CustomUpload Connect Disconnect

Source Web site: D:\Corporate\Projects\GoScan\CustomUpload

Name	Status	Date Modified
App_Code		
App_Data		
Service.asmx	Unchanged	7/30/2008 7:54 AM
Service.cs	Unchanged	7/14/2009 7:51 PM
web.config	Unchanged	7/14/2009 12:48 PM

Remote Web site: V:\CustomUpload

Name	Status	Date Modified
App_Code		
App_Data		
Service.asmx	Unchanged	7/30/2008 7:54 AM
Service.cs	Unchanged	7/14/2009 7:51 PM
web.config	Unchanged	7/14/2009 12:48 PM

Use web publishing to deploy your web services

Last refresh: 7/14/2009 7:51 PM

Show deleted files since the last copy operation

Status:

Copy from Source Web site to Remote Web site is finished. Completed at 7/14/2009 19:51:59. View Log...

Properties

D:\Corporate\Projects\GoScan\Custo

Always Start When True

Full Path: D:\Corporate\Projects\GoScan\CustomUpload

Opened URL: file:///D:/Corporate/Projects/GoScan/CustomUpload

Port number: 3970

Use dynamic ports: True

Virtual path: /CustomUpload

Solution Explorer - D:\...CustomUpload

Solution 'CustomUpload' (1 project)

- D:\...CustomUpload
- App\_Code
- App\_Data
- Service.asmx
- Service.cs
- web.config

Always Start When Debugging

Start the local Web server even when not the startup project

Solution Explorer Class View

Ready

# Create a virtual directory

Create a virtual directory (e.g. CustomUpload) and make sure execute permissions are set correctly!

The screenshot shows the 'CustomUpload Properties' dialog box with the following settings:

- Virtual Directory:  A directory located on this computer
- Local path: C:\bits\CustomUpload
- Script source access:  Script source access
- Read:  Read
- Write:  Write
- Directory browsing:  Directory browsing
- Log visits:  Log visits
- Index this resource:  Index this resource
- Application name: CustomUpload
- Starting point: <GoScan - 16674>\Custo...
- Execute permissions: Scripts and Executables
- Application pool: SharePoint - moss.goscan.com18

# Use MOSS server to host webservices

The screenshot shows a remote desktop session titled "appsvr - Remote Desktop". On the left is the Windows Explorer tree view showing the server's file system structure, including folders like "GoScan - 16674" and "CustomUpload". The main window is a web browser displaying the "Service Web Service" page. The page title is "Service" and the URL is "http://moss.goscan.com:16674/CustomUpload/service.aspx?op=Upload". The page content includes a "Service" header, a link to a complete list of operations, and sections for "Upload", "Test", "SOAP 1.1", and "SOAP 1.2". A callout bubble with a blue border and white background contains the text "Use your MOSS server to host your webservices!". Red arrows point from the bubble to the "Service" header and the "Upload" section. The browser's status bar at the bottom shows "Done" and "Internet".

Service

Click [here](#) for a complete list of operations.

**Upload**

**Test**

The test form is only available for requests from the local machine.

**SOAP 1.1**

The following is a sample SOAP 1.1 request and response. The **placeholders** shown need to be replaced with actual values.

```
POST /CustomUpload/service.aspx HTTP/1.1
Host: moss.goscan.com
Content-Type: text/xml; charset=utf-8
Content-Length: length
SOAPAction: "http://tempuri.org/Upload"

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" x
  <soap:Body>
    <Upload xmlns="http://tempuri.org/">
      <goScanXML>string</goScanXML>
    </Upload>
  </soap:Body>
</soap:Envelope>
```

HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-Length: length

<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" x
 <soap:Body>
 <UploadResponse xmlns="http://tempuri.org/" />
 </soap:Body>
</soap:Envelope>

**SOAP 1.2**

The following is a sample SOAP 1.2 request and response. The **placeholders** shown need to be replaced with actual values.

```
POST /CustomUpload/service.aspx HTTP/1.1
```

- Paul Smietan
  - [psmietan@goscan.com](mailto:psmietan@goscan.com)
  - 909-744-8595
- Mike Stuhley
  - [mstuhley@goscan.com](mailto:mstuhley@goscan.com)
  - 949-829-5822